Capital Conquest: Stock Saga

Student No.	22012112
Name	이새진
E-mail	dltowls000@gmail.com

[Revision history]

Revision date	Version #	Description	Author
29/03/2024	1.00	First Draft	이새진
05/08/2024	1.10	Changed The Topic	이새진
06/11/2024	1.10	Conceptualization 수정	이새진
06/12/2024	1.10	Analysis 수정 및 Design 수정	이새진

= Contents =

1. Introduction

1.1 Summary

최근 들어 많은 사람들이 주식 투자에 관심을 가지게 되면서, 실제 투자를 하기 전에 연습해볼 수 있는 가상 주식 게임에 대한 수요가 늘고 있다. 이는 주식 투자에 대한 진입 장벽을 낮추고, 초보 투자자들이 안전하게 경험을 쌓을 수 있는 기회를 제공한다는 점에서 긍정적으로 평가받고 있다. 그러나, 현재 시장에 제공되고 있는 가상 주식 게임들은 구현이 잘되어 있지 않거나 접근성이 낮은 경우가 많아, 사용자들의 만족도가 높지 않은 편이다. 이러한 문제점을 해결하고자 "Capital Conquest: Stock Saga"라는 가상 주식 게임을 개발하게 되었다.

"Capital Conquest: Stock Saga"는 가상의 기업 정보와 뉴스 기사를 바탕으로 주가 변동이 이루어지는 시뮬레이션 게임이다. 단순히 주식을 매수하고 매도하는 기능을 넘어, Part-Time Job(아르바이트) 기능과 다양한 아이템 요소를 추가하여 게임의 재미 요소를 높였다. 또한, 봇들과의 순위 경쟁을 통해 동기부여를 제공함으로써, 사용자들의 몰입도를 높이고자 하였다.

"Capital Conquest: Stock Saga"는 실제 투자를 하기 전 연습할 수 있는 가상 주식 게임으로, 주식 투자에 대한 이해도를 높이고 재미를 제공하는 것을 목표로 한다. 이를 통해 주식 시장에 대한 진입 장벽을 낮추고 투자 입문자들의 경험을 쌓는 데 도움을 줄 것으로 기대된다.

1.2 Goals

본 보고서에서는 가상주식게임 시스템의 설계를 다루고자 한다. 이를 위해 다음과 같은 UML 다이어그램을 활용하여 시스템의 핵심 기능과 구조를 설계할 것이다. Class Diagram에선 CCSS의 시스템을 구성하는 주요 클래스와 그들 간의 관계를 보여줄 것이다. 이를 통해 시스템의 구조와 핵심 기능을 파악할 수 있다. Sequence Diagram에선 사용자의 주요 행동, 예를 들어 주식 매수/매도, 상점에서의 아이템 구매 등의 동작 흐름을 표현할 것이다. 이를 통해 시스템의 동작 메커니즘을 이해할 수 있다. State Machine Diagram에선 CCSS 내 시스템의 동작을 상태(State)와 상태 간의 전이(Transition)으로 표현하여 시스템의 동작을 직관적으로 이해할 수 있을 것이다.

이러한 설계 과정을 통해 CCSS 시스템의 구조와 동작을 체계적으로 정의하고, 향후 구현 및 확장을 위한 기반을 마련하고자 한다. 이를 통해 CCSS를 구현함으로써, 사용자들이 재미를 느낌과 동시에 주식과 친해질 수 있도록 한다.

2. Class diagram

2.1 Connect

Attributes
- gameName: JLabel 게임의 영어 이름을 표시하는 라벨 - gameNameKr: JLabel 게임의 한국어 이름을 표시하는 라벨 - pressKeyText: JLabel 키 입력을 안내하는 텍스트 라벨 - allPanel: JPanel 전체 패널
Methods
+ Connect() 생성자에서 initComponents() 호출 - initComponents(): void 각 컴포넌트를 초기화 및 레이아웃 설정 + keyPressed(e: KeyEvent): void 키 이벤트를 통해 창 닫기

2.2 SetNickname

Attributes
- infoText: JLabel 닉네임 입력 안내 텍스트 - titleText: JLabel 타이틀 텍스트 - allPanel: JPanel 전체 패널 - nameField: JTextField 닉네임 입력 필드
Methods
+ SetNickname() 생성자에서 initComponents() 호출 - initComponents(): void 각 컴포넌트를 초기화 및 레이아웃 설정 - jTextField1ActionPerformed(evt: ActionEvent): void 닉네임 입력 필드에서 엔터 키를 눌렀을 때 이벤트 처리

2.3 Client

Attributes
- sector: Sector 주식 정보를 관리하는 섹터 객체 - sectors: List<Sector> 주식 섹터 목록 - user: User 사용자 정보를 저장하는 객체 - stockContent: StringBuilder 주식 정보를 HTML 형식으로 저장하는 객체 - now: LocalDateTime 현재 날짜 및 시간 - formatter: DateTimeFormatter 날짜 및 시간을 포맷하는 객체 - formattedDateTime: String 포맷된 현재 날짜 및 시간 문자열 - countNewsMin: int 신문 갱신 타이머의 분 단위 - countNewsSec: int 신문 갱신 타이머의 초 단위 - stockMin: int 주식 갱신 타이머의 분 단위 - stockSec: int 주식 갱신 타이머의 초 단위 - running: boolean 타이머의 실행 상태를 나타내는 플래그 - uiManager: UIManager UI 관리를 위한 객체 - assets: Assets 사용자의 자산 정보를 관리하는 객체 - ranking: Ranking 랭킹 정보를 관리하는 객체 - bot: BotSystem 봇 시스템을 관리하는 객체 - newsLabel: JLabel 신문 내용을 표시하는 라벨 - newsManager: News 뉴스 관리를 위한 객체 - series: TimeSeries 주가 시간 열 데이터 - chart: JFreeChart 주가 차트를 표시하는 객체 - reloadStock: JButton 주가 갱신 버튼 - rankingButton: JButton 랭킹 버튼 - reloadNews: JButton 신문 갱신 버튼 - buyStock: JButton 주식 매수 버튼 - sellStock: JButton 주식 매도 버튼 - selectStock: JToggleButton 주식 분야 선택 토글 버튼 - stockComboBox: JComboBox<String> 주식 목록 콤보 박스 - shopButton: JButton 상점 버튼 - partTimeButton: JButton 알바 버튼 - assetsButton: JButton 자산 버튼 - corporateButton: JButton 기업 정보 버튼 - showTime: JLabel 현재 시간 및 주가 갱신 시간 표시 라벨

- showNewsTime: JLabel 신문 갱신 시간 표시 라벨 - stockPrice: JLabel 주가 정보를 표시하는 라벨 - allPanel: JPanel 전체 패널 - optionPanel: JPanel 옵션 패널 - optionPanel2: JPanel 옵션 패널 - graphPanel: JPanel 그래프 패널 - subPanel: JPanel 서브 패널 - newsPanel: JPanel 뉴스 패널 - chartPanel: JPanel 주가 차트 패널 - newsTextPanel: JPanel 뉴스 텍스트 패널
Methods
+ Client() Client 생성자 + connecting(client: Client): void 클라이언트 연결 및 초기화 작업 수행 + readStocksFromFile(filename: String, sectors: List<Sector>): void 파일에서 주식 데이터를 읽어와 섹터 목록에 추가 + addStockToSector(sectorName: String, stock: Stock, sectors: List<Sector>): void 주식을 섹터에 추가 - initComponents(client: Client): void GUI 컴포넌트를 초기화 + main(args: String[]): void 프로그램 진입점 + changeStockContent(): void 주식 내용을 업데이트 - reloadStockActionPerformed(evt: ActionEvent): void 주가 갱신 버튼 클릭 시 이벤트 처리 - buyStockActionPerformed(evt: ActionEvent): void 주식 매수 버튼 클릭 시 이벤트 처리 - sellStockActionPerformed(evt: ActionEvent): void 주식 매도 버튼 클릭 시 이벤트 처리 - shopButtonActionPerformed(evt: ActionEvent): void 상점 버튼 클릭 시 이벤트 처리 - partTimeButtonActionPerformed(evt: ActionEvent): void 알바 버튼 클릭 시 이벤트 처리 - assetsButtonActionPerformed(evt: ActionEvent): void 총 자산 버튼 클릭 시 이벤트 처리 - reloadNewsActionPerformed(evt: ActionEvent): void 신문 갱신 버튼 클릭 시 이벤트 처리

- selectStockActionPerformed(evt: ActionEvent): void 주식 분야 선택 버튼 클릭 시 이벤트 처리 - rankingButtonActionPerformed(evt: ActionEvent): void 랭킹 버튼 클릭 시 이벤트 처리 - corporateButtonActionPerformed(evt: ActionEvent): void 기업 정보 버튼 클릭 시 이벤트 처리 - updateTimerLabel(): void 신문 갱신 시간 라벨 업데이트 - updateStockLabel(): void 주가 갱신 시간 라벨 업데이트 - stopTimerThread(): void 타이머 스레드 중지 - stockComboBoxActionPerformed(evt: ActionEvent): void 주식 콤보 박스 선택 이벤트 처리 - showUpdatingPopup(message: String): JDialog 갱신 중 팝업 표시 - showCompletedPopup(message: String): void 갱신 완료 팝업 표시 - updateNews(): void 뉴스 업데이트 - applyNewsEffect(news: String, isPositive: boolean): void 뉴스로 인한 주가 변동률 적용 - createDataset(): XYDataset 데이터셋 생성 - createChart(dataset: XYDataset): JFreeChart 차트 생성 - updateChart(): void 차트 업데이트

2.4 Client - TimerThread

Attributes
- sectors: List<Sector> 섹터 리스트 저장 - newsManager: News 뉴스 관리 객체
Methods
+ TimerThread(sectors: List<Sector>, newsManager: News) 타이머 스레드를 생성 및 실행 + run(): void 타이머 스레드 실행 및 주가와 뉴스 갱신 및 시간, 봇들의 돈 업데이트

2.5 Sector

Attributes
- name: String 섹터 이름 - stocks: List<Stock> 섹터에 속한 주식 목록
Methods
+ Sector(name: String) 생성자에서 섹터 이름을 설정하고 주식 목록을 초기화 + getName(): String 섹터 이름을 반환 + setName(name: String): void 섹터 이름을 설정 + getStocks(): List<Stock> 섹터에 속한 주식 목록을 반환 + addStock(stock: Stock): void 주식을 섹터에 추가 + updateStockPrices(sector: Sector): Sector 섹터에 속한 주식의 가격을 업데이트 + resetPrice(stock: Stock): Stock 상장 폐지로 인한 주식 가격을 초기값으로 재설정

2.6 Stock

Attributes
- name: String 주식 이름 - price: int 현재 주식 가격 - prevPrice: int 이전 주식 가격 - changedPrice: int 주식 가격 변동량 - stat: boolean 주식 가격 상승 여부 - notChange: boolean 주식 가격 변동 여부 - delisting: boolean 주식 상장 폐지 여부 - priceHistory: LinkedList<PriceRecord> 주식 가격 변동 기록 - newsMultiplier: double 뉴스에 의한 주식 가격 변동 배율
Methods
+ Stock(name: String, price: int) 주식 이름과 가격을 받아 주식 생성 및 가격 기록을 추가 + getName(): String 주식 이름 반환 + setName(name: String): void 주식 이름 설정 + getPrice(): int 현재 주식 가격 반환

+ getPrevPrice(): int 이전 주식 가격 반환 + getChangedPrice(): int 주식 가격 변동량 반환 + setChangedPrice(): void 주식 가격 변동량 설정 + setPrevPrice(prevPrice: int): void 이전 주식 가격 설정 + setPrice(price: int): void 현재 주식 가격 설정하고 기록 + addPriceToHistory(price: int): void 주식 가격 기록 추가 + getPriceHistory(): List<PriceRecord> 주식 가격 변동 기록 반환 + setStat(stat: boolean): void 주식 가격 상승 여부 설정 + setNotChange(notChange: boolean): void 주식 가격 변동 여부 설정 + setDelisting(delisting: boolean): void 주식 상장 폐지 여부 설정 + getNotChange(): boolean 주식 가격 변동 여부 반환 + getStat(): boolean 주식 가격 상승 여부 반환 + getDelisting(): boolean 주식 상장 폐지 여부 반환 + applyPositiveNewsEffect(): void 긍정적 뉴스 효과 적용 + applyNegativeNewsEffect(): void 부정적 뉴스 효과 적용 + removeNewsEffect(): void 뉴스 효과 제거 + getNewsMultiplier(): double 뉴스에 의한 주식 가격 변동 배율 반환 + setNewsMultiplier(newsMultiplier: double): void 뉴스에 의한 주식 가격 변동 배율 설정

2.7 Stock - PriceRecord

Attributes
- prevPrice: final int 주식의 가격 - time: final LocalDateTime 주식의 가격이 변동된 시간
Methods
+ PriceRecord(price: int, time: LocalDateTime) 주식 가격 및 가격 변동 시간 저장 + getPrice(): 기록된 가격 반환 + getTime(): 변동된 시간 반환

2.8 Assets

Attributes
- user: User 사용자의 정보를 저장하는 객체 - assetsContent: StringBuilder 사용자의 자산 정보를 HTML 형식으로 저장하는 객체 - sectors: List<Sector> 사용자가 보유한 주식의 분야 정보 - itemDialog: JDialog 사용자가 보유한 아이템 표시 - assetsInfo: JLabel 자산 정보를 표시하는 라벨 - assetsPanel: JPanel 자산 레이아웃을 위한 패널 - showItemsButton: JButton 보유 아이템을 볼 수 있는 버튼
Methods
+ Assets(user: User, sectors: List<Sector>) 생성자에서 initComponents() 호출 - initComponents(): void 각 컴포넌트 초기화 + setUser(user: User): void 사용자 설정 및 사용자 정보 업데이트 + getUser(): User 현재 사용자 반환 + setAssets(): void 자산 정보 구성 및 라벨 업데이트 + updateUserInfo(): void 화면에 사용자 정보 업데이트 + displayItemInventory(): void 사용자의 아이템 목록을 표시 - showItemsButtonActionPerformed(evt: ActionEvent): void 보유 아이템 버튼 클릭 시 액션 수행 - useItem(itemName: String): void 아이템 사용 및 사용자 데이터 업데이트 - addRandomStocks(quantity: int): void 랜덤 주식을 지정된 수량만큼 사용자에게 추가 - getRandomSector(): Sector 섹터 목록에서 랜덤 섹터 반환

2.9 BotSystem

Attributes
- user: User 사용자 정보를 저장하는 객체 - botName: ArrayList<String> 봇의 이름을 저장하는 리스트 - botMoney: ArrayList<Integer> 봇의 돈을 저장하는 리스트
Methods
+ BotSystem() 생성자에서 loadBotData() 호출 + setUser(user: User): void 사용자 정보를 설정 + loadBotData(filename: String): void 파일에서 봇 데이터를 로드 + getBotNames(): ArrayList<String> 봇의 이름 목록을 반환 + getBotMoney(): ArrayList<Integer> 봇의 돈 목록을 반환 + setBotMoney(): void 봇의 돈을 랜덤으로 증가 또는 감소시키고, 0 이하로 내려가면 사용자 돈에서 값을 설정

2.10 CorporateInfo

Attributes
- client: Client 클라이언트 정보를 저장하는 객체 - sectors: List<Sector> 주식 분야 목록 - techName: ArrayList<String> 기술 분야 기업 이름 목록 - techEx: ArrayList<String> 기술 분야 기업 설명 목록 - artName: ArrayList<String> 예술 분야 기업 이름 목록 - artEx: ArrayList<String> 예술 분야 기업 설명 목록 - gameName: ArrayList<String> 게임 분야 기업 이름 목록 - gameEx: ArrayList<String> 게임 분야 기업 설명 목록 - selectedCompanies: ArrayList<String> 선택된 기업 이름 목록 - jButton1: JButton 뒤로 가기 버튼 - selectSector: JComboBox<String> 분야 선택 콤보 박스 - selectStock: JComboBox<String> 주식 선택 콤보 박스 - titleText: JLabel 타이틀 텍스트 라벨 - infoText: JLabel 기업 설명 텍스트 라벨 - allPanel: JPanel 전체 패널 - titlePanel: JPanel 타이틀 패널 - infoPanel: JPanel 정보 패널
Methods
+ CorporateInfo(client: Client, sectors: List<Sector>) 생성자에서 기업 정보를 로드하고 initComponents() 호출 - readFile(file: String, name: ArrayList<String>, ex: ArrayList<String>): void 파일에서 기업 정보를 읽어와 이름과 설명 목록에 추가 - initComponents(): void 각 컴포넌트를 초기화 + jComboBox1ActionPerformed(evt: ActionEvent): void 분야 선택 콤보 박스의 액션 이벤트 처리 + jComboBox2ActionPerformed(evt: ActionEvent): void 주식 선택 콤보 박스의 액션 이벤트 처리 + jButton1ActionPerformed(evt: ActionEvent): void 뒤로 가기 버튼 클릭 시 이벤트 처리

2.11 News

Attributes
- positiveNews: List<String> 긍정적인 뉴스 목록 - negativeNews: List<String> 부정적인 뉴스 목록
Methods
+ News() 생성자에서 긍정적인 뉴스, 부정적인 뉴스 목록 초기화 + loadNewsFromFile(filename: String): void 파일에서 뉴스 데이터를 읽어와 파일 이름에 따라 각 뉴스 목록에 추가 + getPositiveNews(): List<String> 긍정적인 뉴스 목록 반환 + getNegativeNews(): List<String> 부정적인 뉴스 목록 반환 + getRandomNews(count: int): List<String> 랜덤으로 뉴스 목록에서 지정된 개수 만큼 뉴스 반환

2.12 PartTime

Attributes
- client: Client 클라이언트 정보를 저장하는 객체 - countMin: int 알바 횟수 추가까지 남은 시간의 분 단위 - countSec: int 알바 횟수 추가까지 남은 시간의 초 단위 - running: boolean 타이머 실행 상태를 나타내는 플래그 - user: User 사용자 정보를 저장하는 객체 - ex: ArrayList<String> 알바 설명을 저장하는 리스트 - pokemon: ArrayList<String> 포켓몬 도감 문제를 저장하는 리스트 - pokemonAnswer: ArrayList<String> 포켓몬 도감 정답을 저장하는 리스트 - word: ArrayList<String> 단어 문제를 저장하는 리스트 - wordAnswer: ArrayList<String> 단어 정답을 저장하는 리스트 - arth: ArrayList<String> 산술 연산 문제를 저장하는 리스트 - arthAnswer: ArrayList<Integer> 산술 연산 정답을 저장하는 리스트 - timer: Timer 타이머 객체 - timeLimitSeconds: int 제한 시간 (초 단위) - random: int 랜덤 인덱스 값 - partMessage: StringBuilder 알바 결과 메시지를 저장하는 객체 - backButton: JButton 뒤로 가기 버튼 - startButton: JButton 알바 시작 버튼 - jComboBox1: JComboBox<String> 알바 유형 선택 콤보 박스 - titleText: JLabel 타이틀 텍스트 라벨 - haveCount: JLabel 알바 가능 횟수 표시 라벨 - countTimer: JLabel 알바 횟수 추가까지 남은 시간 표시 라벨 - exText: JLabel 알바 설명 텍스트 라벨 - answerTimer: JLabel 제한 시간 표시 라벨 - allPanel: JPanel 전체 패널 - titlePanel: JPanel 타이틀 패널 - textPanel: JPanel 텍스트 패널 - answerText: JTextField 정답 입력 텍스트 필드
Methods
+ PartTime(client: Client) 생성자에서 파일을 로드하고 initComponents() 호출 - loadFile(filename: String, list: ArrayList<String>): void 파일에서 데이터를 읽어와 리스트에 추가

- loadArthFile(filename: String, list: ArrayList<Integer>): void 파일에서 산술 연산 정답 데이터를 읽어와 리스트에 추가
+ setUser(user: User): void 사용자 정보를 설정하고 알바 가능 횟수 업데이트
- updateStartButton(): void 알바 시작 버튼 상태 업데이트
+ timerThread.start(): Thread 타이머 스레드 실행
+ initComponents(): void 각 컴포넌트를 초기화
- jComboBox1ActionPerformed(evt: ActionEvent): void 알바 유형 선택 콤보 박스의 액션 이벤트 처리
- backButtonActionPerformed(evt: ActionEvent): void 뒤로 가기 버튼 클릭 시 이벤트 처리
- jButton2ActionPerformed(evt: ActionEvent): void 알바 시작 버튼 클릭 시 이벤트 처리
- updateTimerLabel(): void 알바 횟수 추가까지 남은 시간 라벨 업데이트
- stopTimerThread(): void 타이머 스레드 중지
- partTimeEx(index: int): void 알바 설명을 업데이트
- poketmonStart(): void 포켓몬 도감 알바 시작
- wordStart(): void 단어 맞추기 알바 시작
- arthStart(): void 산술 연산 알바 시작
- checkAnswer(): void 포켓몬 도감 및 단어 맞추기 알바의 정답 체크
- containsAlphabets(text: String): boolean 텍스트에 알파벳이 포함되어 있는지 확인
- checkArthAnswer(): void 산술 연산 알바의 정답 체크
- startTimer(): void 제한 시간 타이머 시작
- updateTimer(): void 제한 시간 라벨 업데이트

2.13 PartTime - TimerThread

Attributes
Methods
+ run(): void: 타이머 실행 로직, 시간 경과를 체크하고 타이머 레이블 업데이트

2.14 Ranking

Attributes
- user: User 사용자 정보를 저장하는 객체 - bot: BotSystem 봇 시스템을 관리하는 객체 - map: HashMap<String, Integer> 사용자 및 봇의 이름과 자산을 저장하는 맵 - botName: ArrayList<String> 봇의 이름 목록 - botMoney: ArrayList<Integer> 봇의 자산 목록 - sortList: List<Map.Entry<String, Integer>> 정렬된 순위 리스트 - sortedMap: LinkedHashMap<String, Integer> 정렬된 순위를 저장하는 맵 - now: LocalDateTime 현재 날짜 및 시간 - formatter: DateTimeFormatter 날짜 및 시간을 포맷하는 객체 - formattedDateTime: String 포맷된 현재 날짜 및 시간 문자열 - rank: int 순위를 나타내는 변수 - curTime: JLabel 현재 시간을 표시하는 라벨 - nameText: JLabel 이름을 표시하는 라벨 - moneyText: JLabel 자산을 표시하는 라벨 - titleText: JLabel 타이틀을 표시하는 라벨 - timePanel: JPanel 시간 표시 패널 - rankPanel: JPanel 순위 표시 패널
Methods
+ Ranking(user: User, bot: BotSystem) 생성자에서 사용자와 봇 정보를 설정하고 initComponents() 호출 + setRanking(user: User, bot: BotSystem): void 사용자와 봇의 순위를 계산하고 화면에 표시 - calculateTotalAssets(user: User): int 사용자의 총 자산을 주식의 금액과 함께 계산 - initComponents(): void 각 컴포넌트를 초기화하고 레이아웃 설정

2.15 SelectOrder

Attributes
- client: Client 클라이언트 정보를 저장하는 객체 - sectors: List<Sector> 섹터 목록 - user: User 사용자 정보를 저장하는 객체 - isSellMode: boolean 매도 및 매수 모드 여부를 나타내는 플래그 - userStockMap: Map<String, List<String>> 사용자가 보유한 주식을 분야별로 정리한 맵 - backButton: JButton 뒤로 가기 버튼 - selectSector: JComboBox<String> 분야 선택 콤보 박스 - selectStock: JComboBox<String> 주식 선택 콤보 박스 - allPanel: JPanel 전체 패널 - contextField: JTextField 수량 입력 필드 - sectorButton: JToggleButton 분야 선택 토글 버튼 - sellButton: JToggleButton 매도 모드 선택 토글 버튼 - buyButton: JToggleButton 매수 모드 선택 토글 버튼
Methods
+ SelectOrder(client: Client, sectors: List<Sector>) 생성자에서 클라이언트와 섹터 목록을 설정하고 initComponents() 호출 + setUser(user: User): void 사용자 정보를 설정 - initComponents(): void 각 컴포넌트를 초기화 - jToggleButton1ActionPerformed(evt: ActionEvent): void 분야 버튼 클릭 시 섹터 콤보 박스를 보이도록 설정 - jComboBox1ActionPerformed(evt: ActionEvent): void 분야 선택 시 주식 콤보 박스를 업데이트 - jComboBox2ActionPerformed(evt: ActionEvent): void 주식 선택 시 수량 입력 필드를 활성화 - jButton1ActionPerformed(evt: ActionEvent): void 뒤로 가기 버튼 클릭 시 이벤트 처리 - jTextField1ActionPerformed(evt: ActionEvent): void 수량 입력 필드에서 엔터 키를 눌렀을 때 이벤트 처리 - jToggleButton2ActionPerformed(evt: ActionEvent): void 매수 버튼 클릭 시 이벤트 처리 - jToggleButton4ActionPerformed(evt: ActionEvent): void 매도 버튼 클릭 시 이벤

트 처리
- checkComboBoxSelection(): void 주식 선택 여부에 따라 수량 입력 필드를 활성화
- updateComboBoxes(): void 콤보 박스를 업데이트
- updateSellComboBox(): void 매도 모드에서 주식 콤보 박스를 업데이트
- getSectorByStock(stock: Stock): Sector 주식에 해당하는 섹터를 반환
- buyStock(quantity: int): void 주식을 매수
- sellStock(quantity: int): void 주식을 매도
- printUserPortfolio(): void 사용자가 보유한 주식 목록을 출력

2.16 Shop

Attributes
- client: Client 클라이언트 정보를 저장하는 객체 - countMin: int 상점 갱신까지 남은 분 - countSec: int 상점 갱신까지 남은 초 - running: boolean 타이머 스레드 실행 여부 - user: User 사용자 정보를 저장하는 객체 - items: ArrayList<String[]> 상점에서 판매하는 아이템 목록 - backButton: JButton 뒤로 가기 버튼 - buy1: JButton 첫 번째 아이템 구매 버튼 - buy2: JButton 두 번째 아이템 구매 버튼 - buy3: JButton 세 번째 아이템 구매 버튼 - titleText: JLabel 상점 타이틀 텍스트 - refreshTimer: JLabel 상점 갱신 타이머 라벨 - item1Name: JLabel 첫 번째 아이템 이름 라벨 - item2Name: JLabel 두 번째 아이템 이름 라벨 - item3Name: JLabel 세 번째 아이템 이름 라벨 - item1Ex: JLabel 첫 번째 아이템 설명 라벨 - item2Ex: JLabel 두 번째 아이템 설명 라벨 - item3Ex: JLabel 세 번째 아이템 설명 라벨 - allPanel: JPanel 전체 패널 - item1Panel: JPanel 첫 번째 아이템 패널 - item2Panel: JPanel 두 번째 아이템 패널 - item3Panel: JPanel 세 번째 아이템 패널 - titlePanel: JPanel 타이틀 패널 - item1NamePanel: JPanel 첫 번째 아이템 이름 패널 - item2NamePanel: JPanel 두 번째 아이템 이름 패널 - item3NamePanel: JPanel 세 번째 아이템 이름 패널 - item1ExPanel: JPanel 첫 번째 아이템 설명 패널 - item2ExPanel: JPanel 두 번째 아이템 설명 패널 - item3ExPanel: JPanel 세 번째 아이템 설명 패널
Methods
+ Shop(client: Client) 생성자에서 클라이언트 설정, 아이템 로드 및 초기화 + setUser(user: User): void 사용자 정보를 설정

- initComponents(): void 각 컴포넌트를 초기화하고 레이아웃 설정
- jButton1ActionPerformed(evt: ActionEvent): void 뒤로 가기 버튼 클릭 시 이벤트 처리
- updateTimerLabel(): void 타이머 라벨 업데이트
- stopTimerThread(): void 타이머 스레드 중지
- loadItems(): void 파일에서 아이템 정보를 로드
- refreshItems(): void 아이템을 갱신하고 버튼을 활성화
- jButton2ActionPerformed(evt: ActionEvent): void 첫 번째 아이템 구매 버튼 클릭 시 이벤트 처리
- jButton3ActionPerformed(evt: ActionEvent): void 두 번째 아이템 구매 버튼 클릭 시 이벤트 처리
- jButton4ActionPerformed(evt: ActionEvent): void 세 번째 아이템 구매 버튼 클릭 시 이벤트 처리
- purchaseItem(itemIndex: int, button: JButton): void 아이템 구매 로직 처리

2.17 UIManager

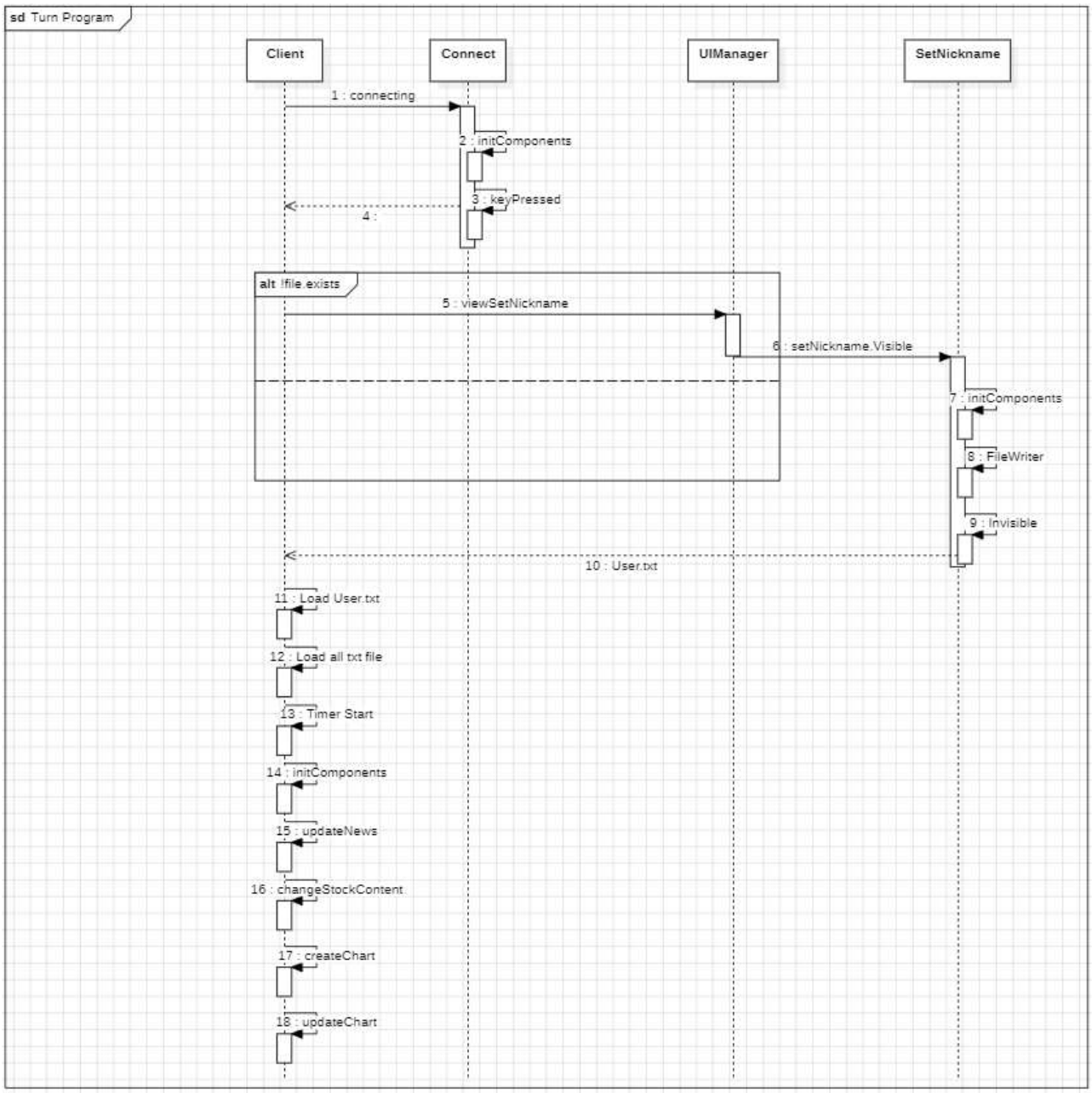
Attributes
- shop: Shop Shop 객체 - client: Client Client 객체 - selectOrder: SelectOrder SelectOrder 객체 - corporateInfo: CorporateInfo CorporateInfo 객체 - partTime: PartTime PartTime 객체 - connect: Connect Connect 객체 - assets: Assets Assets 객체 - setNickname: SetNickname SetNickname 객체 - user: User User 객체 - ranking: Ranking Ranking 객체 - sectors: List<Sector> 섹터 목록
Methods
+ UIManager(client: Client, sectors: List<Sector>) UIManager 생성자 + getConnect(): Connect Connect 객체 반환 + getSetNickName(): SetNickName SetNickName 객체 반환 + viewShop(): void Shop 화면으로 전환 + setShop(user: User): void Shop 객체 설정 + viewOrder(): void Order 화면으로 전환 + setOrderUser(user: User): void SelectOrder 객체 설정 + viewCorporateInfo(): void CorporateInfo 화면으로 전환 + viewPartTime(): void PartTime 화면으로 전환 + setAssets(assets: Assets): void Assets 객체 설정 + viewAssets(): void Assets 화면으로 전환 + setRanking(ranking: Ranking): void Ranking 객체 설정 + viewRanking(): void Ranking 화면으로 전환 + viewSetNickname(): void SetNickname 화면으로 전환

2.18 User

Attributes
- name: String 사용자 이름 - money: int 사용자 보유 금액 - partTimeCount: int 사용자 알바 가능 횟수 - stockPortfolio: Map<Stock, Integer> 보유 주식 저장 - stockPrices: Map<Stock, Double> 주식 가격 기록 - itemInventory: Map<String, Integer> 아이템 인벤토리 저장
Methods
+ User(file: String) 생성자, 파일에서 사용자 데이터를 읽어와 초기화 + setMoney(money: int): void 사용자 보유 금액 설정 + setPartTimeCount(partTimeCount: int): void 사용자 알바 가능 횟수 설정 + getName(): String 사용자 이름 반환 + getMoney(): int 사용자 보유 금액 반환 + getPartTimeCount(): int 사용자 알바 가능 횟수 반환 + addStock(stock: Stock, quantity: int, price: int): void 주식 추가 + removeStock(stock: Stock, quantity: int): void 주식 제거 + getStockQuantity(stock: Stock): int 특정 주식의 보유 수량 반환 + getStockPortfolio(): Map<Stock, Integer> 보유 주식 반환 + getStockPrice(stock: Stock): double 특정 주식의 가격 반환 + addItem(itemName: String): void 아이템 추가 + getItemInventory(): Map<String, Integer> 아이템 인벤토리 반환

3. Sequence diagram

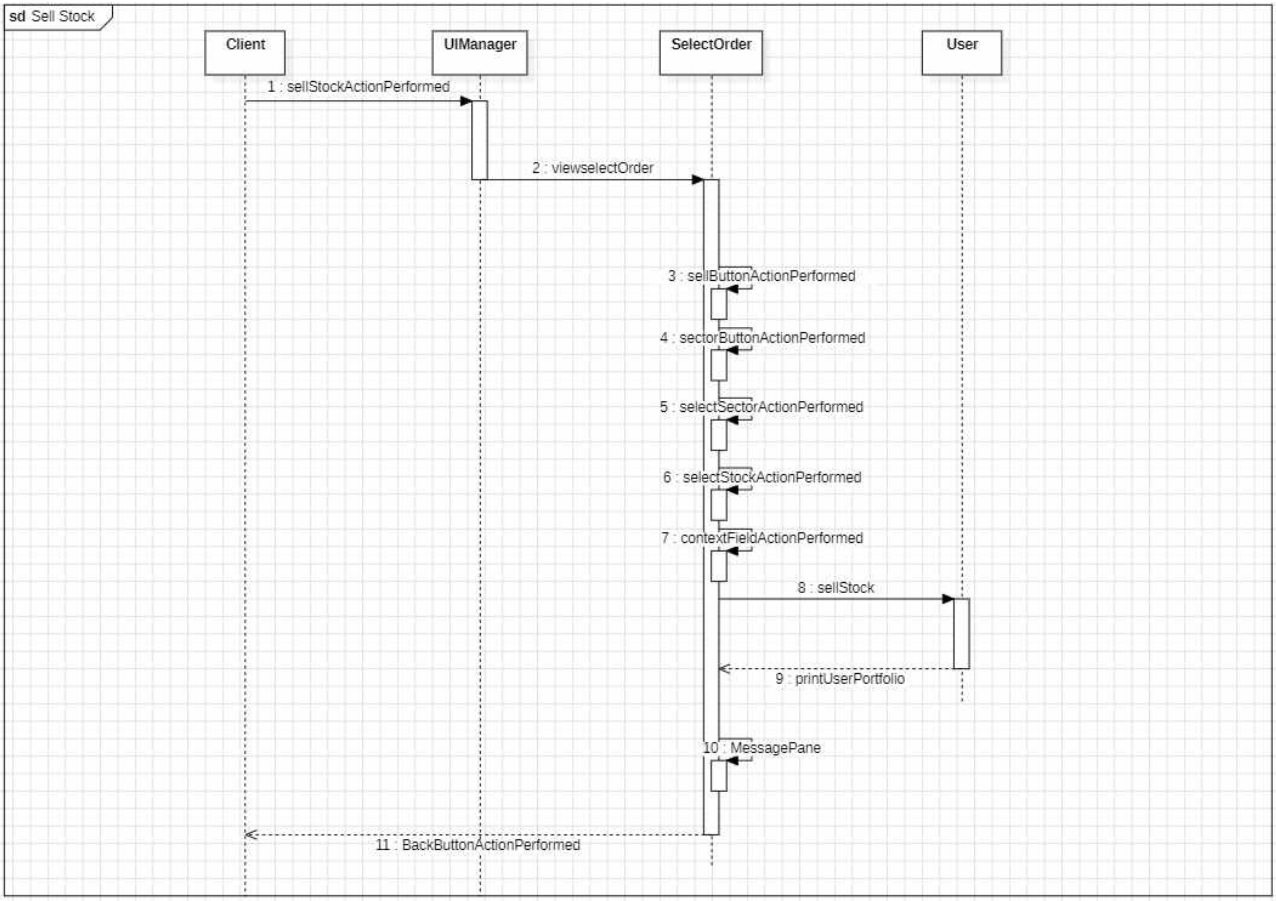
3.1 Turn Program



<그림 3-1> Turn Program Sequence Diagram

이 Sequence Diagram은 프로그램이 시작될 때의 Sequence를 나타낸다. Client가 제일 처음 실행되고, connecting 함수를 통해 Connect GUI를 나타내며, 아무 키 입력을 받아 닉네임 저장 여부에 따라 SetNickName으로 이동하거나 다시 Client에서 유저 파일 불러오기, 기타 파일 모두 불러오기, 타이머 시작, 레이아웃 초기화, 뉴스 출력, 주가 출력, 주가에 대한 그래프 출력을 실행한다.

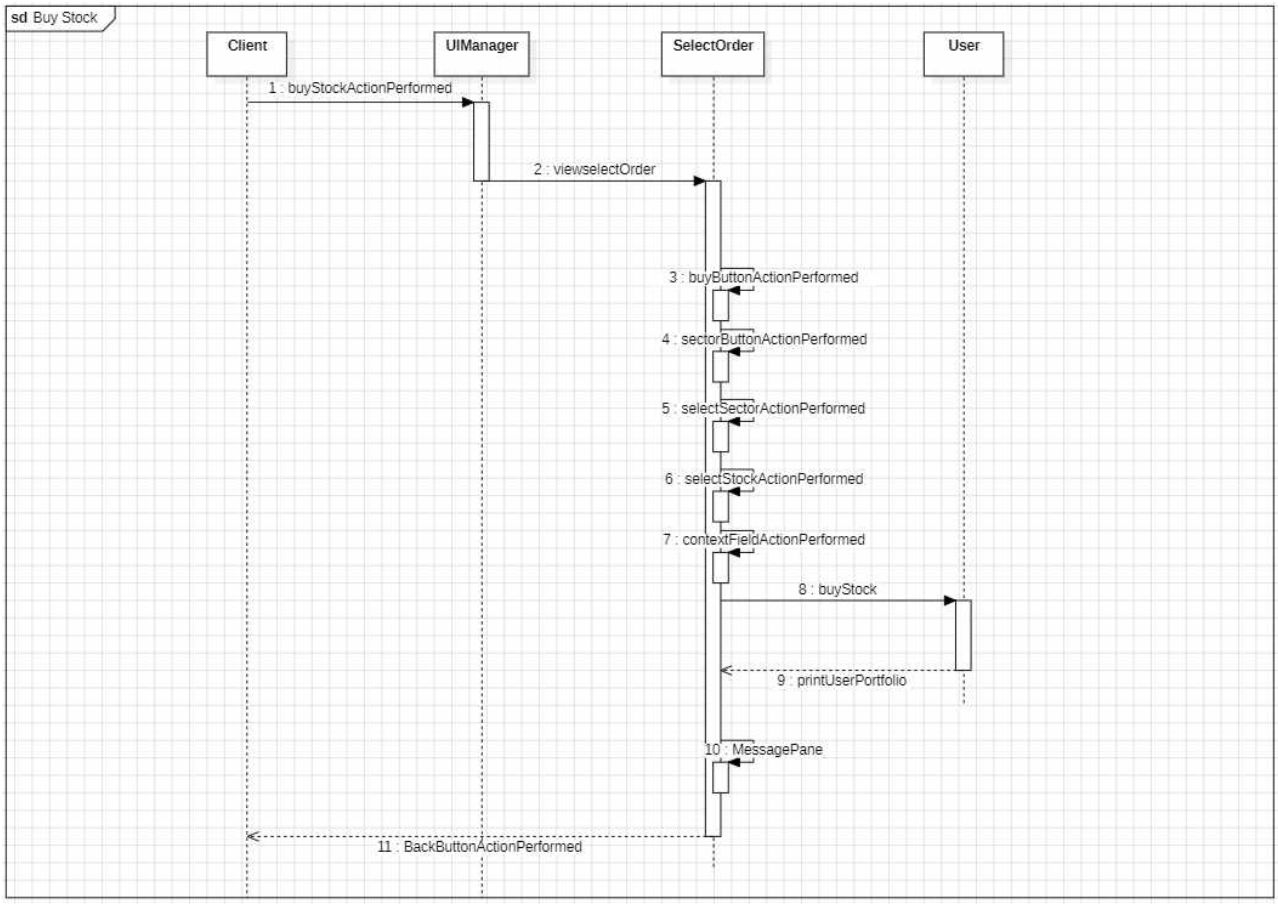
3.2 Sell Stock



<그림 3-2> Sell Stock Sequence Diagram

Client에서 매도 버튼을 누를 경우 ActionPerformed의 핸들러를 통해 UIManager에서 viewSelectOrder 함수를 호출한다. 이후, 유저는 시장가 매도 버튼, 분야 버튼, 분야 선택, 주식 선택, 수량 입력을 통해 주식을 판매하고, 판매한 주식에 대한 정보를 User에게 업데이트한 후, User는 판매한 주식에 대한 정보를 반환한다. 해당 반환을 통해 팝업창을 출력하고 BackButton을 누를 시, Client로 돌아간다.

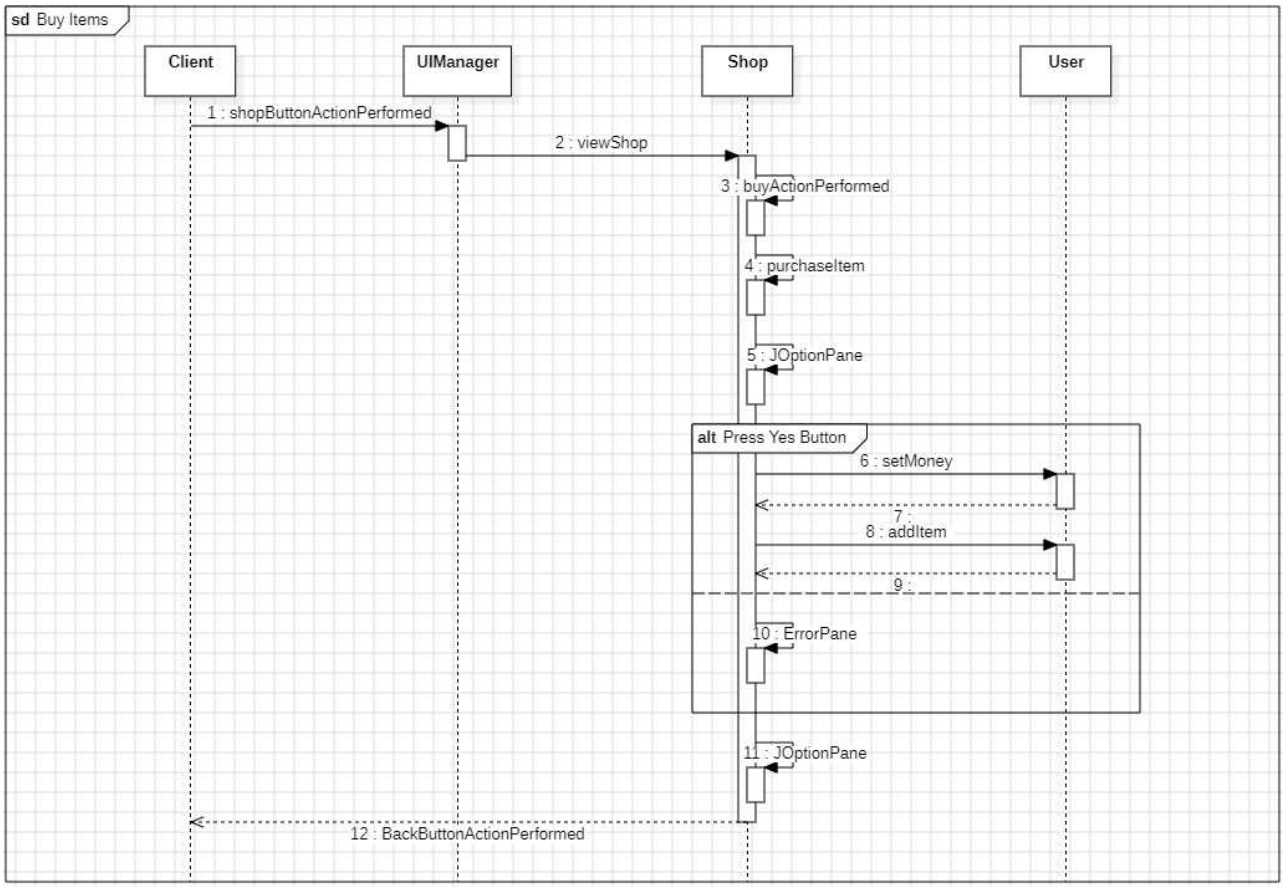
3.3 Buy Stock



<그림 3-3> Buy Stock Sequence Diagram

Client에서 매수 버튼을 누를 경우 ActionPerformed의 핸들러를 통해 UIManager에서 viewSelectOrder 함수를 호출한다. 이후, 유저는 시장가 매수 버튼, 분야 버튼, 분야 선택, 주식 선택, 수량 입력을 통해 주식을 구매하고, 구매한 주식에 대한 정보를 User에게 업데이트한 후, User는 구매한 주식에 대한 정보를 반환한다. 해당 반환을 통해 팝업창을 출력하고 BackButton을 누를 시, Client로 돌아간다.

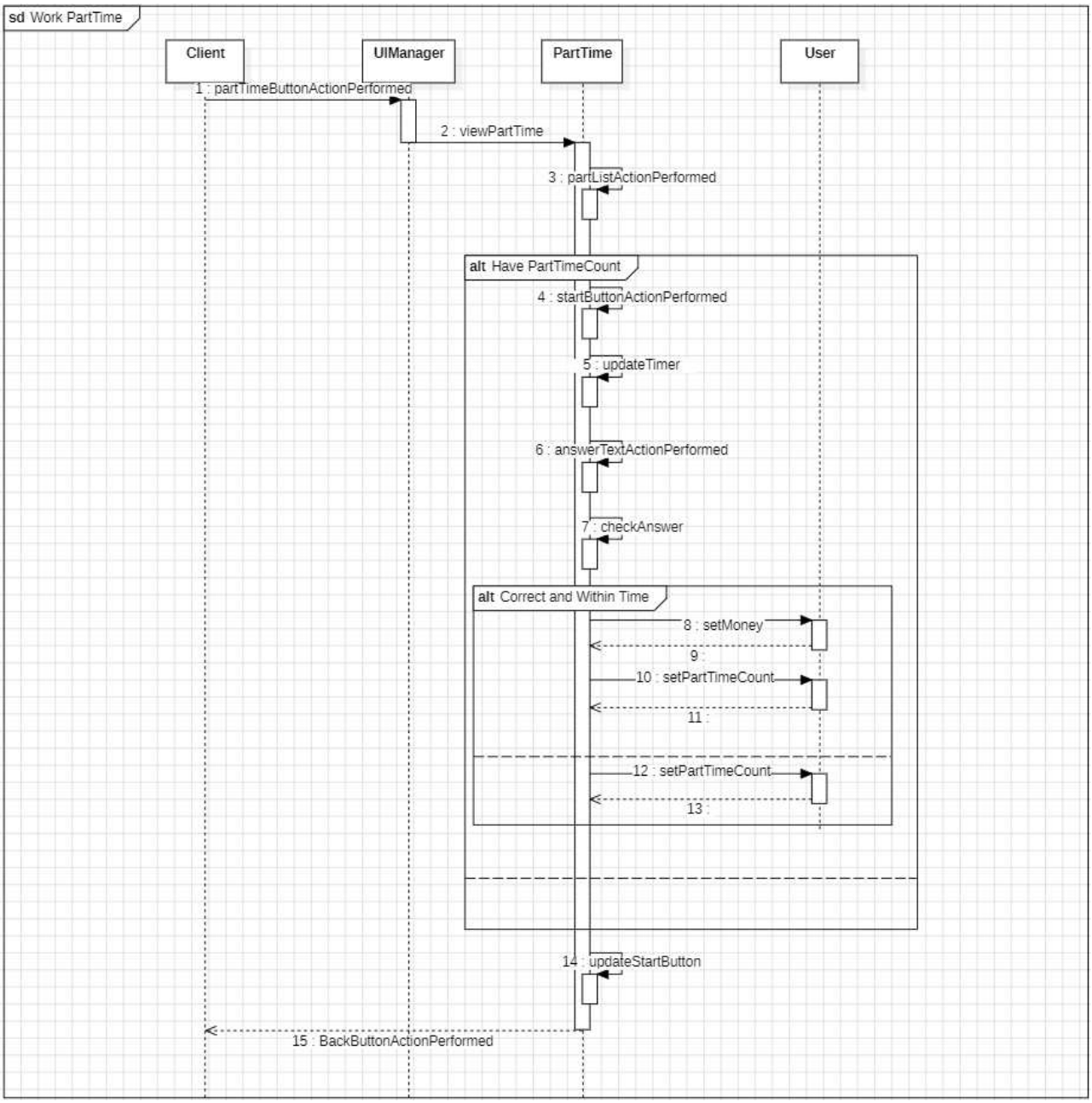
3.4 Buy Items



<그림 3-4> Buy Items Sequence Diagram

해당 Sequence Diagram은 사용자가 상점에서 아이템을 구매하는 Sequence를 나타낸다. Client에서 상점 버튼을 누를 경우, 핸들러에 의해 상점 UIManager를 통해 상점 GUI를 출력하고, 구매 버튼을 누르면 팝업창을 출력하며, Yes 버튼을 누를 경우, 돈이 충분하다면 아이템 구매가 성공적으로 이뤄지며, 구매로 인해 감소한 돈과 구매한 아이템에 대한 정보를 User로 넘겨주고, 성공적인 구매에 대한 팝업창을 띄우며, 돈이 부족한 경우, 해당 경우에 맞는 팝업창을 띄운다. 이후 BackButton을 누르면 Client로 돌아간다.

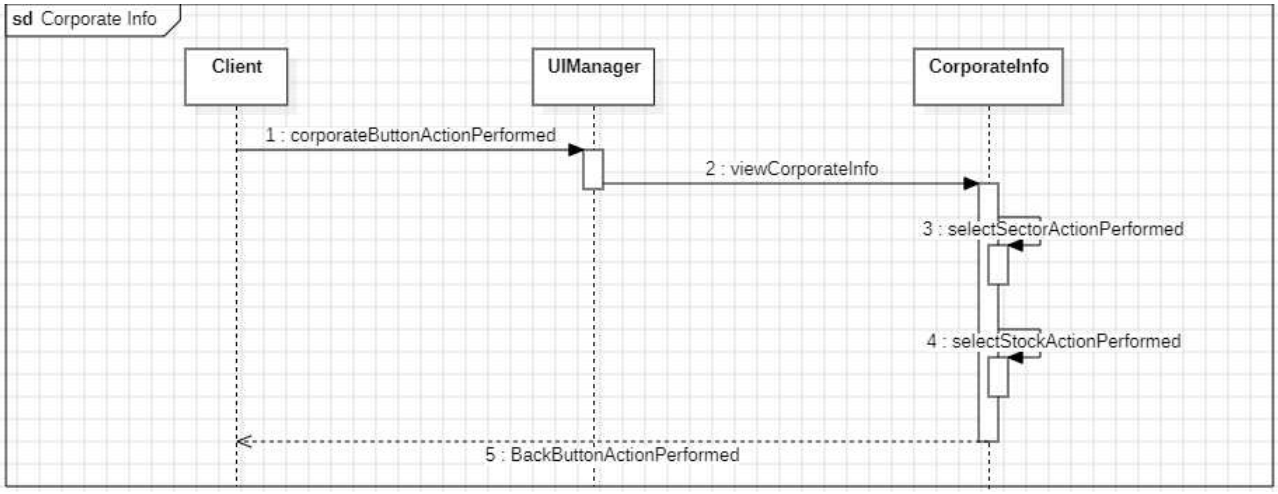
3.5 Work PartTime



<그림 3-5> Work PartTime Sequence Diagram

알바에 대한 Sequence를 담고 있다. 마찬가지로, Client에서 알바 버튼을 누를 경우, 핸들러에 의해 UIManager에서 viewPartTime을 호출하여 알바 GUI를 출력한다. 이후, 알바를 선택하고 시작 버튼을 누르면 제한 시간 타이머가 시작되고, answerText의 핸들러를 통해 사용자가 입력하고 엔터 키를 누르면, checkAnswer를 호출하여, 정답일 때와 오답일 때와 더불어 제한 시간 초과가 관련하여 증가한 돈 및 알바 횟수 감소 또는 알바 횟수만 감소하고, 현재 알바 가능 횟수에 맞게 시작 버튼의 상태를 업데이트하며, 뒤로가기 버튼을 누를 경우, Client로 돌아간다.

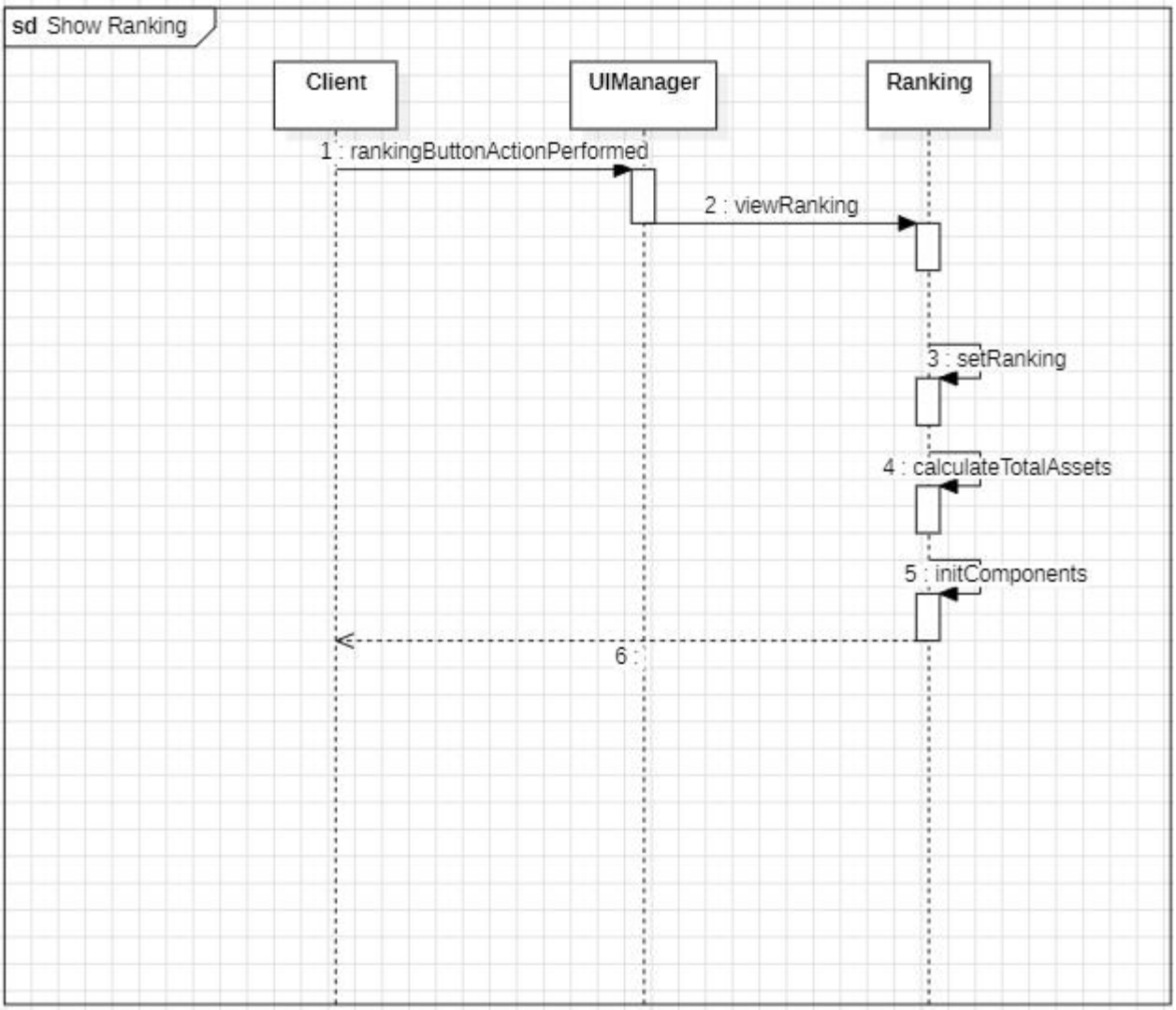
3.6 Corporate Info



<그림 3-6> Corporate Info Sequence Diagram

이 Sequence Diagram은 Client와 CorporateInfo 간의 상호작용 과정을 보여준다. 먼저 사용자가 Client에서 기업 정보 버튼을 누르면 핸들러에 의해 UIManager의 viewCorporateInfo를 호출하여 GUI를 출력한다. 이후, 유저는 분야와 기업을 선택하면 기업에 대한 설명을 확인할 수 있으며, BackButton을 누르면 Client로 돌아간다.

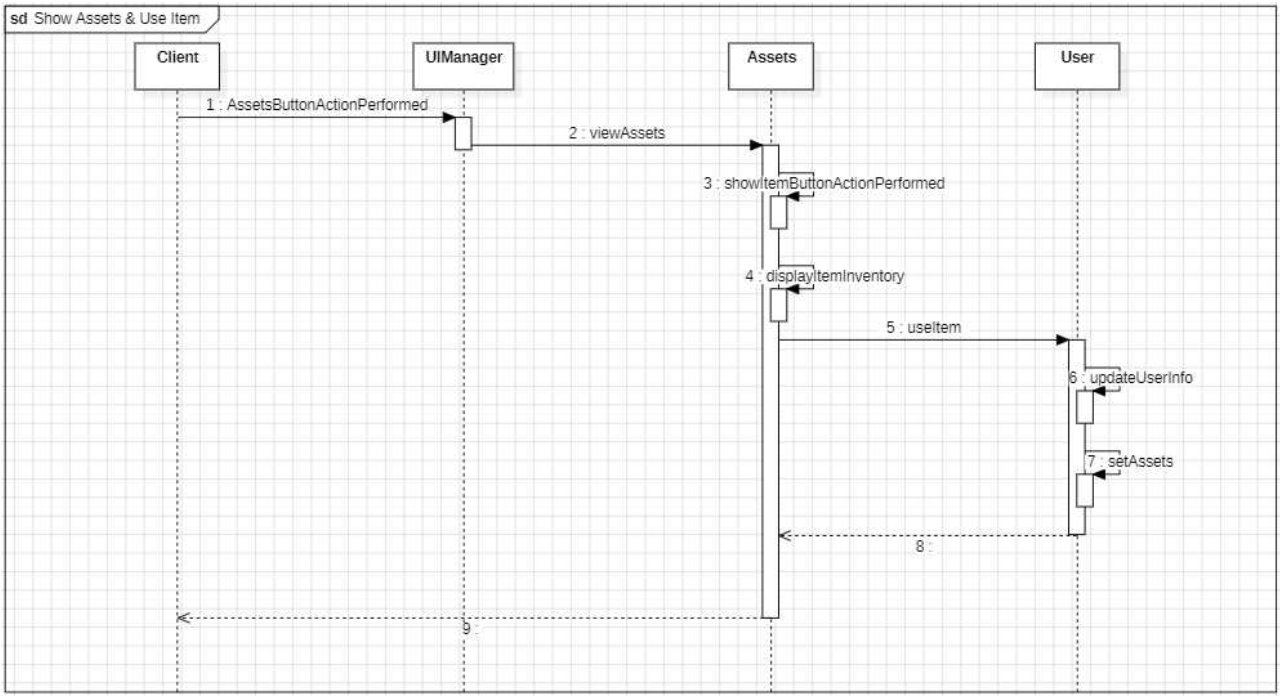
3.7 Show Ranking



<그림 3-7> Ranking Sequence Diagram

Client에서 랭킹 버튼을 누르면 핸들러에 의해 UIManager의 viewRanking이 호출되어 랭킹을 보여주는 GUI가 추력된다. GUI가 출력될 때, setRanking을 통해 랭킹을 선정하기 시작하며, calculateTotalAssets를 호출하여 유저가 가지고 있는 돈과 주식의 가격을 통해 랭킹을 정한다. 내림차순으로 정렬하여 1등부터 10등까지의 유저를 포함한 봇들의 랭킹을 보여준다.

3.8 Show Assets & Use Item

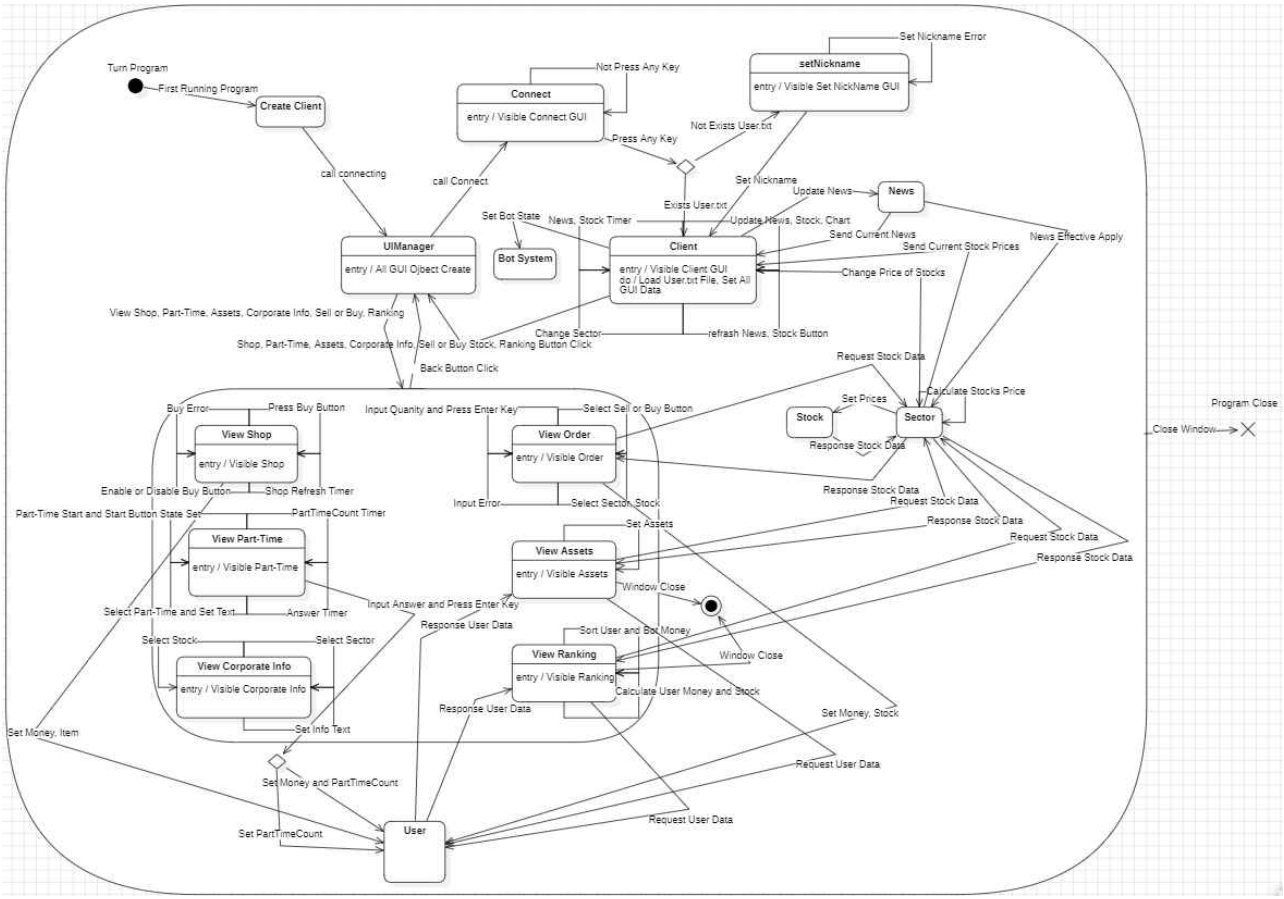


<그림 3-8> Show Assets & Use Item Sequence Diagram

이는 Client가 총 자산 버튼을 클릭했을 때, 핸들러에 의해 UIManager의 viewAssets가 호출되어 총 자산을 나타내는 Assets GUI가 출력된다. 해당 화면에서 보유 아이템 보기 버튼을 누르면 showItemButtonActionPerformed 핸들러로 인해 displayItemInventory가 호출되고, 유저가 보유하고 있는 아이템을 보여준다.

만약 아이템을 사용할 경우, User의 정보에서 사용한 아이템과 총 자산을 업데이트 한다.

4. State machine diagram



사용자가 프로그램을 실행하면, Client 객체를 만들고, Connect 화면으로 이동한다. Connect화면에서 아무 키나 입력을 하게되면 닉네임이 존재하는지 여부를 확인하고, 있으면 나머지 GUI 객체를 생성 및 데이터를 로드, 없으면 닉네임 설정하는 화면으로 이동하여 닉네임을 입력한 후, 나머지 GUI 객체 생성 및 데이터 로드한다. GUI 객체 생성 및 데이터 로드가 끝나면 Client 화면으로 이동하고, 해당 화면에서 각 버튼을 누르면 해당 버튼의 기능에 맞는 화면으로 이동하거나 팝업창이 나타난다. 주식 매수와 매도, 아이템 구매 및 PartTime 완료 상태에 따라 User의 데이터를 업데이트 하며, Client에서 버튼을 누르면 이동한다는 같은 로직을 가지고 있는 State끼리 사진과 같이 표현하였으며, 만약 Assets와 Ranking의 창을 닫을 경우, 해당 State만 종료되지만, 나머지 State의 창을 닫을 경우 프로그램이 종료된다.

5. Implementation requirements

5.1 S/W Requirements

- Implemetation Language : JAVA
- OS : Windows
- OS Version : 7.0 이상

6. Glossary

Term	Description
ActionPerformed	Java Swing 및 AWT (Abstract Window Toolkit)에서 사용되는 이벤트 핸들러 인터페이스 사용자가 버튼을 클릭하거나 메뉴 항목을 선택하는 등의 작업을 수행했을 때 발생하는 이벤트를 처리하는 데 사용
GUI	"Graphical User Interface"의 약자로, 그래픽 사용자 인터페이스를 나타냄
Diagram	정보나 데이터를 시각적으로 나타내거나 표현하기 위해 사용되는 도표, 그림을 가리킴

7. References

강의 자료 : 영남대학교 오픈소스SW설계 Behavior Modeling I, II